Design of a Speech Anger Recognition System on Arduino Nano 33 BLE Sense

Dania Maryam Waqar
Electrical and Computer Engineering Department.
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
daniawaqar@gmail.com

Teddy Surya Gunawan
[1]ECE Department, International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
[2]School of Electrical Engineering and Telecommunications, UNSW
Sydney, NSW 2052, Australia
tsgunawan@iium.edu.my

Malik Arman Morshidi
Electrical and Computer Engineering Department
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
mmalik@iium.edu.my

Mira Kartiwi
Information Systems Department
International Islamic University Malaysia
53100 Kuala Lumpur, Malaysia
mira@iium.edu.my

2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA) | 978-1-7281-7523-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICSIMA50015.2021.9526323

***Abstract*—Speech Emotion Recognition (SER) has gained growing popularity due to its wide applications in almost every available field. Past work has been done on large-scale processing boards with a variety of extracted features. Many different methods have been proposed for the SER system but have lacked aspects in either size, complexity, or recognition accuracy. Due to limitations on size and resources, designing a system that can overcome these drawbacks becomes imperative. The solution is to design a system that is small, accurate, and economical. Using Tiny Machine Learning and SER is the best solution since it can be done on a small-scale and relatively high emotion recognition rate. This paper presents past work, the hardware, software, and the SER prototype's field design, focusing on detecting the variations of the Anger emotion. A simple and optimum CNN architecture was developed for Arduino Nano 33 BLE Sense implementation. Prototype validation showed that our system could detect not angry, about to be angry, and angry emotions.**

Keywords—speech emotion recognition, anger recognition, Arduino, Tensor Flow, TinyML.

I. INTRODUCTION

Speech Emotion Recognition (SER) is a topic that is receiving more attention in recent years due to its range of applications in modern-day technology. From handheld devices (for example, using Siri) to household applications (for example, controlling lights using Amazon Echo), speech recognition was already a much-researched topic. More focus has been placed on human-machine interaction nowadays to aid in gaming, call center applications, and mobile communication [1]. SER is a field that deals with Machine Learning problems to detect the underlying emotion in audio samples [2].

The overall process for SER includes generating an audio signal, extracting its features, and adopting a model or algorithm to classify the emotional state [3]. These three stages and their respective processes and details are covered below. SER can be defined as the classification of a speaker's emotional state based on their speech signal. Although some models and algorithms may add additional emotional states, there are a few universal emotions in every speech corpus: sadness, joy, neutral, and anger [4]. The overall process for Speech Anger Recognition (SAR) is highlighted using the flowchart in Figure 1.

For many years, SER has been done using many different methods, i.e., SVM, HMM, and GMM, while using many different features and methods to extract these features. Many application domains rely on using these systems on large-scale devices while simultaneously using GPU accelerators to speed up processing. Due to this, prices are high, while recognition rates may not always be significant. We can conclude that the main problem lies in the large processing board size, unclear features being extracted leading to a variable recognition rate, and the subsequent elevation of prices.

The audio signals used in experiments are normally based on speech databases (called a corpus) which are either universally recognized or self-made for the project. These databases hold a range of emotions in many different languages that have been acted out and stored for use in SER models and algorithm testing. One vital issue that the SER project has to overcome is determining the set of emotions that would suffice to show the full range of experienced feelings. Since emotion is a very subjective sense, it is not easy to quantify the feelings into words with clear boundaries between them. The second vital matter is designing an appropriate database that has to be as natural as possible. The more naturally flowing pieces of speech available, the more esteemed the experiment would be considered, as there would be a lower probability of incorrect conclusions being determined.

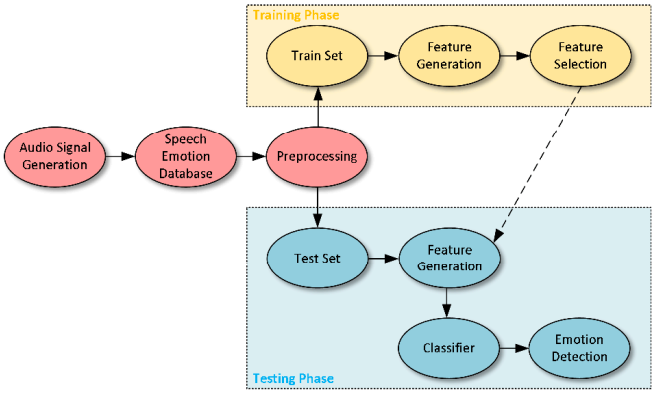

Fig. 1. Typical SER Algorithm

Two major design criteria must be considered: designing the appropriate database and designing the feature vector. The first database design principle is whether or not the emotions in the recording are natural or acted. The experiments nowadays use a given speech corpus to test their models. For example, in the HMM study conducted by [5], they used a

978-1-7281-7523-2/21/$31.00 ©2021 IEEE

Spanish corpus in which actors were hired to simulate emotions by reading the paragraphs of text. However, some may argue that these actors are professionals and may provide a histrionic display of emotions in their speech. To avoid this overcompensation, some may consider natural recordings a more reliable source of data, also known as 'found speech.'

An example of this would be the recordings of a news broadcaster during the crash of Hindenburg, which contained recordings filled with panic and excitement [6]. However, using these recordings may bring up legal and moral complications that may forbid their research purposes. Therefore, the second design criteria to be considered in databases are speaking and their emotional context. Since different people from different cultural backgrounds express the same words differently, it is vital to note the actor's background when analyzing his speech.

The third design attribute for databases lies in the way the speech should be simulated. For example, when professional actors were used, they were given a text to read and could place themselves in the required mental state to produce the correct emotional response. The next design attribute to consider is the number and type of statements used. The number of statements per emotion can vary, although many believe that the number should reflect the current world usage of the emotion. For example, the neutral emotion style should have the greatest number of statements recorded, while, for example, the disgusting emotional style should have a lesser amount since people generally speak more neutrally than a disgusted one. However, some researchers also believe that the distribution should be fair and equal for a more precise experiment to be performed to construct the Berlin corpus [7].

Although many emotion speech databases are available, most of them are not available to researchers due to legal and ethical reasons. For our design, we have decided to use RAVDESS [8] (Ryerson Audio-Visual Database of Emotional Speech and Song), SAVEE (Surrey Audio-Visual Expressed Emotion) [9], and CREMA-D (Crowd-sourced Emotional Multimodal Actors Dataset) [10] databases. The RAVDESS database consists of 24 professional actors (12 male and 12 female) with 7 emotions: Calm, Happy, Sad, Angry, Fearful, Surprise, and Disgust. The audio recordings consist of around 60 trials for every actor, amounting to 1440 recordings, all of 16 bits and 48kHz. The SAVEE database consists of 7 emotions (Anger, Disgust, Fear, Happiness, Sadness, Surprise, and Neutral) performed by 4 male actors, totaling 480 utterances, all of which were sampled at 44.1kHz. Finally, the CREMA database holds 7,442 files from 91 actors, of which 48 are male and 43 are female. All the actors come from various races and ethnicities and speak from a pool of 12 sentences. The database consists of six emotions: Anger, Disgust, Fear, Happy, Neutral, and Sad, all sampled at 48kHz.

II. Comparison of Single Board Computers

Various Single Board Computer (SBC) are listed in Table I. We can see that all the SBCs present viable options to be used in implementing the Deep Learning Algorithm using TensorFlow Lite, except the Arduino UNO, since the framework is not supported there yet. The Beaglebone AI and Jetson Nano can be eliminated from the potential SBC pool due to their heating problems and high cost. The STM32F746G and SparkFun Edge can also be disregarded due to their expense. We can already see much research done between the Raspberry Pi and Arduino Nano 33 BLE Sense when implementing SER using Raspberry Pi. Since this Arduino is cheaper than the Raspberry Pi, and it is an SBC that has not been used with Speech Anger Recognition (SAR) before, we can see the need to explore this option as it would lead to a new SAR system that is cheaper than the one implemented with the Raspberry Pi.

Therefore, in this research, we have chosen the Arduino Nano 33 BLE Sense to implement our project because of its lower price and ease of use due to the extensive libraries and resources available. Although the processing speed is a little slower, this would be compensated by the product's price. Further, the processing speed could be increased with other additions to make it an unnoticeable drawback.

TABLE I. Comparison of Single Board Computers [10, 13, 14]

SBC	Hardware	Software	Cost (Approx.)
Arduino NANO 33 BLE Sense	Advantages: Simple and easy to connect, includes onboard storage, huge community available to help address the bugs faced, includes a microphone and an accelerometer. Disadvantages: Slower CPU and processing capabilities.	Advantages: Many libraries are offered for many different types of hardware, which supports TensorFlow Lite. Disadvantages: Can only run one program at a time.	RM 136
Arduino UNO	Advantages: Simple and easy to connect, not expensive, huge community available to help address the bugs faced. Disadvantages: Slower CPU and processing capabilities	Advantages: Many libraries are offered for many different types of hardware. Disadvantages: Can only run one program at a time, does not support TensorFlow Lite	RM 25
SparkFun Edge	Advantages: Huge community available to help address the bugs faced, faster processing speed.	Advantages: Performs audio analysis on the device itself, supports TensorFlow Lite.	RM 145
ST Microelectronics STM32F746G	Advantages: Can connect to Arduino Uno High processing power	Advantages: Comprehensive software library available. Supports TensorFlow Lite	RM 242
Raspberry Pi	Advantages: Low power architecture, large addressing space Disadvantages: No internal storage	Advantages: Supports many different coding languages, supports TensorFlow Lite Disadvantages: Cannot run on Windows Operating System	RM 155
Beaglebone AI	Advantages: High performance per watt, contains internal storage Disadvantages: Heating problem	Advantages: Supports TensorFlow Lite Disadvantages: Difficult to obtain the latest machine learning libraries	RM 141
NVIDIA's Jetson Nano	Advantages: On-board 128 Core Maxwell GPU – Faster processing speed, large onboard storage of 4GB, has USB 3.0 for faster external storage Disadvantages: Large heat sink	Advantages: Consists of the built-in Jetson Inference library, supports TensorFlow Lite	RM 405

III. Deep Learning In SER Using TinyML

TinyML can be defined as a working neural network model that can run with powers less than 1 mW, leading to a portable and small product (hence the word tiny) to fit into any application that requires compact but complex machine learning algorithms [11] Warden. The way machine learning works is by constructing a system model, which depends on the data that has been input by the programmer. This model construction is called training. After training has occurred, the data which makes our predictions is input into the model, and this process is called inference [11]. TensorFlow Lite was introduced to accommodate projects that are to be executed on a smaller platform. It is designed to run on only a few kB of space, and hence the libraries do not have any operating systems, C or C++ dependency, or any hardware which deals with floating-points, or any memory that can be allocated dynamically [11].

The types of deep learning algorithms that can be implemented on TinyML are CNN, SVM, or a combination known as Transfer Learning [12]. As shown in [12], different emotions were classified using a deep neural network based on Raspberry Pi. The machine learning API used was Keras which uses TensorFlow to define and train the model. The model is a DNN model which consists of many CNN layers, Pooling layers, and Fully Connected (FC) layers. Cascading the convolution layers allows the model to learn more features while simultaneously keeping the filer size small.

IV. Proposed Speech Anger Recognition System

A. Overall Methodology

Overall, the project's methodology is shown in Figure 2. The project aims to train the model using the preprocessed data from which MFCC features are extracted. If the training accuracy is sufficient, the model is tested and converted to TF-Lite format for deployment. When live data is tested, the model automatically extracts MFCC features from the audio input which is then fed into the model. The type of LED to blink would then depend on the probability of anger detected by the model. Using informal evaluation, we found that the confidence level threshold of larger than 0.98 can detect angry, while between 0.55 and 0.98 can detect about to be angry. In addition, less than 0.55 is classified as not angry.

B. Software Design

The block diagram for our proposed software is shown in Figure 3. The human speech signal is the input to our system. Speech contains many emotions which can be detected by its pitch, loudness, or energy. However, we have to feed the speech signal as input into our model to detect these emotions. To feed it in as input, we first need to perform some preprocessing computations to transform the signal into its smoother counterpart and make it easier for the model to extract relevant and dominant features that make emotion recognition more accurate.

1) Preprocessing

We perform preprocessing on the signals by converting each audio file to be of the same length to extract features. Additionally, since audio files from three different databases were being used, they all had to be converted to the same sampling frequency of 16kHz. This sampling frequency has to be chosen since this is the same sampling frequency of the Arduino Nano 33 BLE Sense board. It was done using Pydub on Google Colab [16].

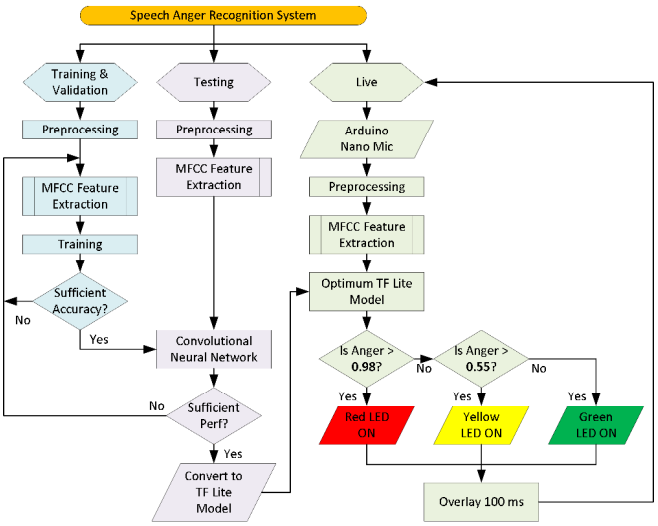

Fig. 2. Flowchart of Anger Recognition System

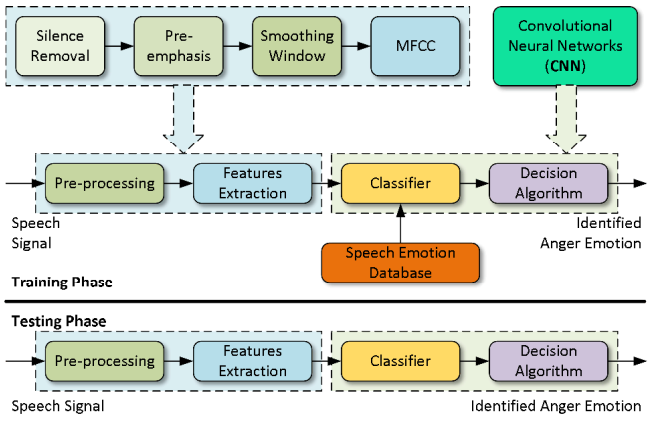

Fig. 3. Proposed Anger Detection Algorithm

2) Train and Test Set

The training and validation set is 80% of the total number of speech samples, 2319 samples. The test set is 10% of the total, amounting to 595 samples.

3) Features Extraction

The one-dimensional signal would first have to be converted into a spectrogram to identify the proper emotion using the Deep Neural Network. The spectrogram is represented as a two-dimensional array. Since our input is now a spectrogram, we would deal with an input tensor with four dimensions. The first dimension acts as a wrapper that contains a single element. The second and third dimensions both symbolize the rows and columns of the spectrogram. The last dimension has a size of 1 and is meant to contain each pixel of the spectrogram.

To create each row of the spectrogram, we would have to run each slice of audio through a Fast Fourier Transform (FFT) algorithm, which would analyze the frequency distribution of the audio signal and create the corresponding arrays. To build a two-dimensional array, however, we would have to integrate the results of the FFT audio slices with each slice overlapping the previous slice by a period of, for example, 10ms. Finally, it results in a spectrogram that is ready to pass into our model.

In our design, we have decided to implement the code using MFCCs. It is the most commonly used feature in speech recognition [13, 15]. The Mel Frequency Cepstral Coefficients help provide us with the short-time power spectrum that shows the vocal tract's shape. Each speech segment is split and then converted into the frequency domain using short-time discrete Fourier Transform. The Mel filter bank is used to compute the sub-band energies. The logarithm is then applied to these energies before passing them through an inverse Fourier Transform to obtain the MFCC values [15].

4) CNN Architecture

Fig. 4 shows the optimum CNN architecture after informal evaluation. In brief, our proposed CNN architecture has two convolutional layers with max-pooling in between. After dropout and flatten layers, the output layer predicts the anger with a confidence level between 0 and 1. In total, there are 4594 trainable parameters for this architecture.

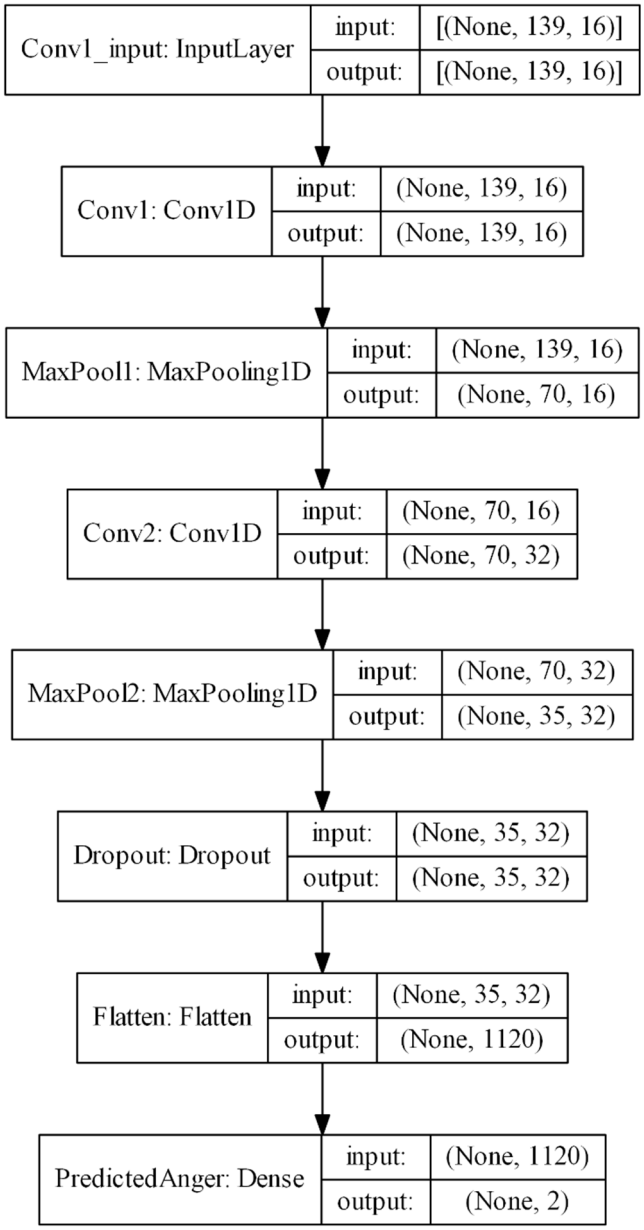

Fig. 4. CNN architecture

Before feeding our MFCCs into the CNN layers, we have to reshape them to be recognized by the input layers. Once this is done, the following layers are used in the architecture. Firstly, the Conv1D layer provides temporal convolution by creating a 1D kernel which convolves with the input layer over a single temporal dimension to produce a 2D tensor of outputs. We have used 16 filters in the first and 32 in the second layer as this combination gives us the highest accuracy. Next, the Max Pooling 1D Layer is responsible for down-sampling the input data by picking the largest value over the spatial window of predefined size (the pool size). Next, we have used the Dropout Layer to set random input units to zero with a frequency of 0.25 (the rate defined) at every step during training. It means that one in every four input units is randomly dropped. The Flatten Layer is used next to flatten the shape of the previous layer's output, while the batch size remains unaffected. Finally, we use the Dense Layer to perform the prediction using the softmax function.

C. Hardware Design

The TensorFlow framework we are using is mainly used to build and train the model. The model that we build is technically an instruction set that informs the interpreter on the method used to transform data for output. The model is to be loaded into memory when it has to be executed by the TensorFlow interpreter. The only issue here is that this interpreter is normally used to run on powerful desktop computers and not on tiny microcontrollers. To run our model on small SBCs, we need to convert the TensorFlow into a TensorFlow Lite format. TensorFlow supports an interpreter with an additional set of tools to allow models to be run on smaller boards. This set of tools are called TensorFlow Lite.

To convert the file into a TensorFlow Lite format, we use a TensorFlow Lite Converter. It would create a C array to hold the file's content, which can then be run on the Arduino Nano 33. With this code, the audio would first be read in through the Arduino's microphone, which would then be processed and compared with the training set data. The corresponding emotion would then be output onto the computer screen. Not only would this solve our problem of figuring out how to deploy the model onto a small board, but it would also allow the model to run faster, hence optimizing performance.

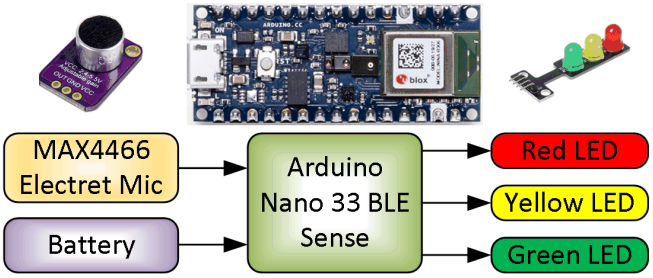

Fig. 5. Hardware Design

The proposed hardware design is shown in Figure 5. We are using Arduino Nano 33 BLE Sense, which has a built-in microphone. Nevertheless, it can be connected to an external microphone, such as the MAX4466 electret microphone. The Arduino board then acts as the model and processes the speech signal powered by a laptop. When the appropriate emotion has been detected, the LED flashes a specific color showing the successful recognition of the emotion. For example, when Angry speech is detected, the Red LED lights up, when an About to Be Angry speech signal is detected, the Yellow LED lights up, and when a Not Angry speech signal is detected, the Green LED lights up.

V. Implementation and Prototype Validation

We implemented our model using TensorFlow Lite. TensorFlow is an open-source machine learning library developed by Google in 2015. TensorFlow allows models to be trained and run on desktops, allowing designers to trade off a larger size for greater functionality. Additionally, since the main interface language used was Python, this allowed a wide range of users to utilize this framework since it is one of the most popular scripting languages that has been used on servers. However, this tradeoff between size and functionality may not always be ideal. For example, to implement models on smaller devices such as iPhone or Android phones, the model's size must be small to keep up speed and performance rates.

To allow models to be run on smaller platforms, Google introduced the TensorFlow Lite library in 2017. The difference between these two frameworks lies in TensorFlow Lite's inability to train models and only run inferences. It is the reason why the training phase would have to be done on a cloud platform first before converting the model into its TensorFlow Lite equivalent. Furthermore, removing some of the lesser-used functionalities from TensorFlow allows the Lite version to fit only a few hundred kilobytes. However, for models to run on even smaller boards, the Google team in 2018 built a specialized version of TensorFlow Lite, which focused only on deploying these models onto embedded platforms. It is how we obtain the current version of TensorFlow Lite that we are using on our chosen Single Board Computer [11].

To implement our model, we would have to ensure that the sensor's input is the same form the model has been trained. Henceforth, this modified data is passed through the same preprocessing techniques as the training data was and input into the model to run the inference. The result of this inference would be data that contains predictions. The implementation is shown in Fig. 6.

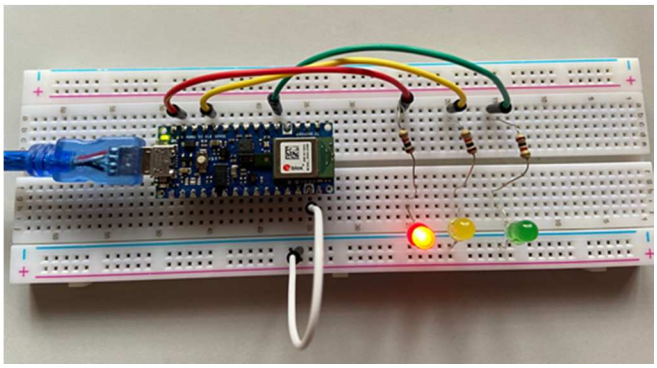

Fig. 6. Implementation of Speech Anger Recognition System

Since it is possible for a sequence of words in a statement to vary in terms of the emotions presented, it is better to practice taking an average of the output over a period of time. It prevents any memory glitches from occurring and produces a more accurate output that only predicts the output based on the dominant emotion since any transient issues would be ignored.

We produced accuracy and cross-entropy graphs to see how the training is going. It can be done using TensorBoard. To see the trends, we have smoothed out the graph. Next, to deploy the model onto the Arduino Nano 33 BLE Sense, we would have to freeze the TensorFlow model. Before that, however, we must first note what exactly is a TensorFlow model. A TensorFlow model comprises mainly two aspects: the weights and biases from training and the weights and biases' results linked with the input to produce the output (shown on the graph). Once the model has been made, freezing occurs. Freezing of a TensorFlow model means we are trying to produce the graph's static representation, which consists of the weights and biases from the training phase. Some of the training arguments and the absolute file path are passed into the code when freezing the model. The resultant is a fully trained TensorFlow model, which can then be converted into a TensorFlow Lite format for SBC deployment, i.e., Arduino Nano 33 BLE Sense.

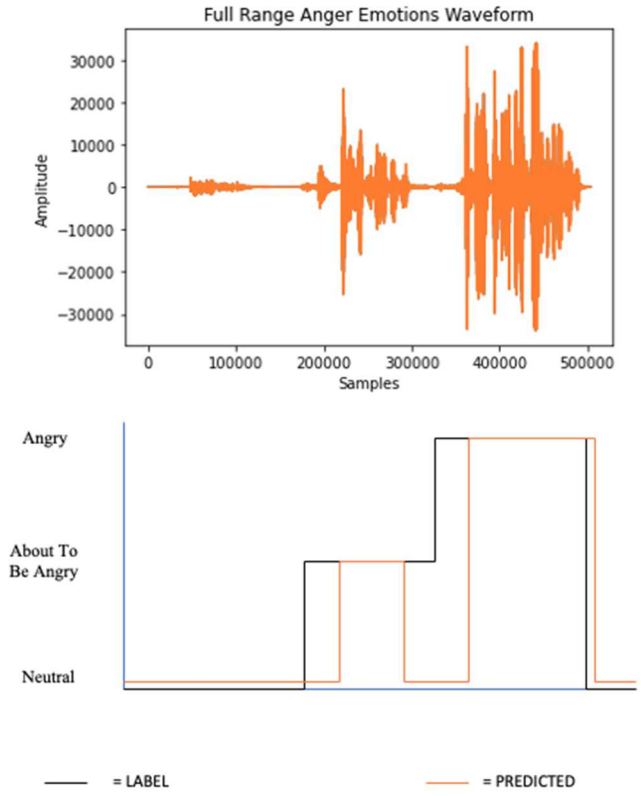

Fig. 7. Prototype Validation using Anger Signal

The prototype validation is shown in Figure 7. The system is able to detect the three conditions, i.e., not angry (green), about to be angry (yellow), and angry (red). In real applications, we can emphasize the second condition, i.e., about to be angry, as it provides an early warning system for various applications, notably in customer service. It is hoped that our proposed system could help the customer service to better handling the customer relationship.

VI. Conclusions

This paper has presented the prototype design and implementation of a speech emotion recognizer. First, it introduces speech emotion recognition and then provides the steps involved in audio generation by also mentioning the databases proposed for the design. Then, the paper provides a summary of related works and possible SBC's to use. Next, an optimum CNN architecture for SCB implementation was proposed. Finally, the proposed hardware and software design is highlighted, along with the proposed implementation. Further research would include the performance evaluation of the developed prototype using various speech signals, different CNN architectures, and different devices.

ACKNOWLEDGMENT

This research was supported by the Ministry of Education Malaysia (MOE) through Fundamental Research Grant Scheme (FRGS) (Ministry Project ID: FRGS/1/2018/ICT01/UIAM/02/1) under Grant (FRGS19-068-0676).

REFERENCES

[1] S. Lalitha, S. Tripathi, D. Gupta, "Enhanced speech emotion detection using deep neural networks," *International Journal of Speech Technology*, 22(3), pp.497-510, 2018.

[2] T. Roy, T. Marwala, S. Chakraverty, "A Survey of Classification Techniques in Speech Emotion Recognition," *Mathematical Methods in Interdisciplinary Sciences*, pp.33-48, 2020.

[3] Y. Pan, P. Shen, L. Shen, "Speech Emotion Recognition Using Support Vector Machine," *International Journal of Smart Home*, 6(2), 2012.

[4] M. Selvaraj, R. Bhuvana, S.P. Karthik, "Human speech emotion recognition," *International Journal of Engineering & Technology*, 8, pp. 311-323, 2016.

[5] A. Nogueiras, A. Moreno, A. Bonafonte, J.B. Mariño, "Speech emotion recognition using Hidden Markov Models", In Seventh European Conference on Speech Communication and Technology, pp. 2679-2682, 2001.

[6] N. Campbell, "Databases of Emotional Speech," in ISCA Tutorial and Research Workshop on Speech and Emotion, pp. 114-121, 2000.

[7] F. Burkhardt, A. Paeschke, M. Rolfes, W.F. Sendlmeier, B. Weiss, "Berlin EmoDB: A database of German emotional speech," in Ninth European Conference on Speech Communication and Technology, 2005.

[8] S.R. Livingstone, F.A. Russo, "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English," PLoS ONE 13(5), 2018. https://doi.org/10.1371/journal.pone.0196391.

[9] P. Jackson, S. Haq, "Surrey Audio-Visual Expressed Emotion (SAVEE) database," University of Surrey, UK, 2014.

[10] H. Cao, D.G. Cooper, M.K. Keutmann, R.C. Gur, A. Nenkova and R. Verma, "CREMA-D: Crowd-Sourced Emotional Multimodal Actors Dataset," *IEEE Transactions on Affective Computing*, vol. 5, no. 4, pp. 377-390, 2014. Available: 10.1109/taffc.2014.2336244

[11] P. Warden, D. Situnayake, "TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers," O'Reilly Media, 2019.

[12] V. Srinivasan, S. Meudt, F. Schwenker, "Deep learning algorithms for emotion recognition on low-power single-board computers," In IAPR Workshop on Multimodal Pattern Recognition of Social Signals in Human-Computer Interaction, pp. 59-70, 2018. https://doi.org/10.1007/978-3-030-20984-1_6

[13] T. M. Wani, T. S. Gunawan, S. A. A. Qadri, M. Kartiwi and E. Ambikairajah, "A Comprehensive Review of Speech Emotion Recognition Systems," in IEEE Access, vol. 9, pp. 47795-47814, 2021, doi: 10.1109/ACCESS.2021.3068045.

[14] T.S. Gunawan, I.R.H. Yaldi, M. Kartiwi, N. Ismail, N.F. Za'bah, H. Mansor, A.N. Nordin, "Prototype Design of Smart Home System using Internet of Things," Indonesian Journal of Electrical Engineering and Computer Science, vol. 7, no. 1, p. 107, 2017. Available: 10.11591/ijeecs.v7.i1.pp107-115

[15] M.F. Alghifari, T.S. Gunawan, M. Kartiwi, "Speech emotion recognition using deep feedforward neural network," Indonesian Journal of Electrical Engineering and Computer Science, 10(2), 554-561, 2018.

[16] T.S. Gunawan, A. Ashraf, B.S. Riza, E.V. Haryanto, R. Rosnelly, M. Kartiwi, & Z. Janin, "Development of video-based emotion recognition using deep learning with Google Colab", TELKOMNIKA, 18(5), 2463-2471, 2020.